ERROR RECOVERY IN A CLIENT/SERVER APPLICATION USING TWO INDEPENDENT SOCKETS FOR COMMUNICATION

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The invention relates generally to the field of computer systems and, more specifically, to a technique for recovering from a single socket failure at a host using two independent sockets.

Description of the Related Art

[0002] Computer systems include host computers that communicate with one anther via a network to run network applications. Such applications include earlier text-based applications that provide remote access to computers, electronic mail, file transfers, news groups and chats, as well as more recent multimedia applications such as the World Wide Web, Internet telephony, video conferencing and audio- and video-on-demand. Typically, software is distributed and run on two or more hosts or end systems to realize the application. For example, for a Web application, browser software on the user's host computer communicates with Web server software in a Web server. The processes running on the different hosts communicate with one another by exchanging messages across a network via sockets. The network applications have application-layer protocols that define the format and order of the messages that are exchanged between the processes. Moreover, the application-layer protocols define what actions to take when a message is transmitted or received. For example, HTTP is an application-layer protocol for the Web that defines how messages are passed between a browser and a Web server. SMTP is an application-layer protocol used for electronic mail.

[0003] In particular, a network application typically includes a client side and a server side. In this case, the application may be referred to as a client/server application. A client side on an end host may communicate with a server side on another host. The client is usually the host that initiates a communication or session with another host. In some cases, a host can implement both the client and server sides of an application. For example, a mail server can run the client side of SMTP for sending mail, as well as the server side of SMTP for receiving mail. In this case, the client-side host opens two sockets – one for sending mail and one for receiving mail. In other cases, a host can implement two sockets that are independent. For example, in client/server applications where communications are initiated by a client and a server independent of each other, two communication sockets at each host can be used to simplify the communication. However, this complicates error recovery when there is a socket failure since each socket can fail independently, without the other socket knowing. For example, it is not known if an individual socket has failed or if the entire connection has been lost. Recovery from these scenarios is also problematic.

BRIEF SUMMARY OF THE INVENTION

[0004] To overcome these and other deficiencies in the prior art, the present invention provides a technique for recovering from a socket failure in a host.

[0005] In a particular aspect of the invention, a method is provided for detecting a failure at a first host having a first socket and a second socket, where the first host's first socket is used for sending requests to a second host's first socket, and receiving responses from the second host's first socket, and the first host's second socket is used for receiving requests from the second host's second socket, and sending responses to the second host's second socket. The method includes detecting when a failure condition occurs at the first host's second socket, and, when the failure condition is detected: attempting to send a communication from the first host's first socket to the second host's first socket; if

the attempt to send succeeds, closing the first host's second socket, then attempting to reconnect the first host's second socket; and, if the attempt to reconnect succeeds, setting an internal state to indicate that normal operation is resumed.

[0006] The above method is used with a server-initiated socket at a client host. Related methods for a client-initiated socket at a client host, and client- and server-initiated sockets at a server host, are also provided. Related host computer systems and program storage devices are also provided.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] These and other features, benefits and advantages of the present invention will become apparent by reference to the following text and figures, with like reference numbers referring to like structures across the views, wherein:

[0008] Fig. 1 illustrates a client process and a server process with send and receive sockets;

[0009] Fig. 2 illustrates two client processes and a server process with send and receive sockets;

[0010] Fig. 3 illustrates a block diagram of a host;

[0011] Fig. 4 illustrates a state diagram for a host recovery process;

[0012] Fig. 5 illustrates a failure recovery process for a client host on its server-initiated receive socket;

[0013] Fig. 6 illustrates a failure recovery process for a client host on its client-initiated send socket;

[0014] Fig. 7 illustrates a failure recovery process for a server host on its server-initiated receive socket; and

[0015] Fig. 8 illustrates a failure recovery process for a server host on its client-initiated send socket.

DETAILED DESCRIPTION OF THE INVENTION

[0016] Fig. 1 illustrates a client process and a server process with send and receive sockets. A computer host 100 runs one or more client processes 120 with a first socket 122 and a second socket 124, while a corresponding computer host 150 runs one or more server processes 170 with a first socket 172 and a second socket 174. Each socket can communicate with the corresponding socket at the other host by sending and receiving data. Here, there are two communication paths 128, 178, which can be independent. One path 128 is between sockets 122 and 172, and the other path 178 is between sockets 124 and 174. In the host 100, the socket 122 is used to send requests to the socket 172 at the host 150, and to receive synchronous responses to those requests. The socket 124 is used by the host 100 to receive requests from the host 150 and send synchronous responses back in response. The socket 122 can be termed a "send" socket or a "client-initiated" socket since a client is the host or process that initiates a session with another host or process, termed the server, in a client-server application. Conversely, the socket 172 can be termed a "receive" socket or a "server-initiated" socket. The socket 172 is used by the server to receive requests from the client and send synchronous responses back. Likewise, the socket 124 can be termed a receive or server-initiated socket, and the socket 174 can be termed a send or client-initiated socket. The socket 174 is used by the server to send requests to the client and receive synchronous responses back in response. The two communication paths 128, 178 can traverse any computer network such as the Internet.

[0017] Fig. 2 illustrates two client processes and a server process with send and receive sockets. Here, the host 100 and an additional host 200 both communicate with host 300. The host 100 has one or more processes 120 that use a send socket 122 to communicate with a receive socket 372 at the host 300 via a communication path 328, and a receive socket 124 to communicate with a send socket 374 at the host 300 via a communication path 330. The host 200 has one or more processes 220 that use a send

socket 222 to communicate with the receive socket 382 at the host 300 via a communication path 348, and a receive socket 224 to communicate with the send socket 384 at the host 300 via a communication path 350. The socket failure recovery process described herein can be used generally at any host that communicates with one or more other hosts. The host 300 implements one or more processes 320 that use the sockets 372, 374, 382 and 384.

[0018] Fig. 3 illustrates a block diagram of a host. The host 400 may be a conventional computer workstation, server, personal computer or other computer system, for instance. The host includes a control 420 that executes software instructions that are stored in a memory 430. Generally, the control 420 may execute any type of computer code devices, such as software, firmware, micro code or the like, to achieve the functionality described herein. Accordingly, a computer program product comprising such computer code devices may be provided in a manner apparent to those skilled in the art. Moreover, the memory 430 is a program storage device that tangibly embodies a program of instructions executable by a machine such as the control 420 to perform a method that achieves the functionality described herein. Such a program storage device may be provided in a manner apparent to those skilled in the art. A display 410 is optionally included with the host 400. A network interface 440 enables the host 400 to communicate with other hosts. For example, the network interface may be any network adapter card used for communicating with other hosts via a communication path 450 and a network 460.

[0019] Fig. 4 illustrates a state diagram for a host recovery process. As described further below, each host may maintain a state relating to the status of its sockets. A "normal" state 480 indicates both sockets are operating normally, and no problems have been detected. An "attempting to recover" state 485 indicates a problem has been detected with at least one of the sockets, and the host is attempting to resolve the problem. For example, a failure may be cause by an operating system error, a lack of

communication at the socket, or removal or failure of a communication medium. A "disconnected" state 490 indicates the problem cannot be resolve, so the host with the problem is disconnecting from the other host. Other states may be maintained if desired, e.g., relating to a specific type of problem that has been detected for a socket, and to a specific corrective measure that is being attempted.

[0020] The invention solves the problem of recovering from a socket error when two sockets are used at a host. In particular, the invention defines a procedure for recovering from a single socket failure that incorporates determining if the other socket has failed before attempting recovery. The process is applied to each socket on both the client and server sides. The result is four independent processes that work to recover from a failure. The four independent processes are described below.

[0021] Fig. 5 illustrates a failure recovery process for a client host on its server-initiated receive socket. For example, the client host, e.g., a first host, may be Host A 100 in Fig. 1, and the server-initiated receive socket may be the socket 124, e.g., a second socket. At block 505, the host listens on the server-initiated receive socket. Listening refers to waiting to receive a communication on a socket. For example, the server listens on the client-initiated or send socket, and the client listens on the server-initiated or receive socket. At block 510, if no problem is detected, the host continues listening. If a problem is detected while listening, the host sets its internal state to indicate it is attempting to recover from the problem (block 515). At block 520, the host determines whether it can communicate on the client-initiated send socket, e.g., a first socket such as socket 122, by attempting to send a communication on the client-initiated send socket. If the host cannot communicate on the send socket, it closes the send socket (block 525) and sets its internal state to indicate it is disconnected from the other host (block 530), e.g., a second host such as Host B 150. If the host can communicate on the send socket, it closes the receive socket, then attempts to reconnect the receive socket (block 532). If the reconnection succeeds (block 540), the host updates its state to indicate it has

resumed normal operation (block 535), and the recovery is successful. If the reconnection fails, the host closes the client-initiated send socket (block 525) and sets internal state to disconnected (block 530).

[0022] Fig. 6 illustrates a failure recovery process for a client host on its client-initiated send socket. For example, the client host, e.g., a first host, may be Host A 100 in Fig. 1, and the client-initiated send socket may be the socket 122, e.g., a first socket. At block 605, the host communicates on the send socket such as by sending a communication to a second host, e.g., Host B 150, or reading a response from the second host. At block 610, if a socket problem is detected while sending a communication to the second host or reading a response from the second host on the send socket, the host checks its internal state. If the internal state indicates the host is attempting to recover from a socket failure (block 615), e.g., on the other, second socket 124, which is the client-side server-initiated receive socket, the host stops and allows the receive socket recovery process to attempt the recovery (block 620). A constraint for the recovery may be imposed such as a time limit or maximum number of tries. If the internal state indicates the host is operating normally, and not attempting to recover, it sets its internal state to indicate that it is attempting to recover (block 625). At block 635, the host attempts to reconnect client-initiated send socket. If the reconnect attempt succeeds (block 640), recovery is successful, and the host sets its internal state to indicate it has resumed normal operation (block 655). If the reconnect attempt fails, the host closes the server-initiated receive socket (block 645) and sets its internal state to indicate that it is disconnected from the second host (block 650).

[0023] Fig. 7 illustrates a failure recovery process for a server host on its server-initiated receive socket. For example, the server host, e.g., a first host, may be Host B 150 in Fig. 1, and the server-initiated receive socket may be the socket 172, e.g., a first socket. At block 705, the host listens on the server-initiated receive socket. At block 710, if no problem is detected, the host continues listening. If a problem is detected

while listening, the host sets its internal state to indicate it is attempting to recover from the problem (block 715). At block 720, the host determines whether it can communicate on the client-initiated send socket, e.g., a second socket such as socket 174, by attempting to send a communication on the send socket. If the host cannot communicate on the send socket, it closes the send socket (block 725) and sets its internal state to indicate it is disconnected from the other host (block 730), e.g., a second host. If the host can communicate on the send socket, it waits for the client host (e.g., Host A 100) to reconnect the receive socket, e.g., socket 172. If the reconnection succeeds before a timeout (block 745), the host updates its state to indicate it has resumed normal operation (block 735), and the recovery is successful. If the reconnection fails, the host closes the client-initiated send socket (block 725) and sets internal state to disconnected (block 730).

[0024] Fig. 8 illustrates a failure recovery process for a server host on its client-initiated send socket. For example, the server host, e.g., a first host, may be Host B 150 in Fig. 1, and the client-initiated send socket may be the socket 174, e.g., a second socket. At block 805, the host communicates on the send socket such as by sending a communication to a second host, e.g., Host A 100, via socket 124, or reading a response from the second host. At block 810, if a socket problem is detected while sending a communication to the second host or reading a response from the second host on the send socket, the host checks its internal state. If the internal state indicates the host is attempting to recover from a socket failure (block 815), e.g., on the other, first socket 172, which is the server-side server-initiated receive socket, the host stops and allows the receive socket recovery process to attempt the recovery (block 820). A constraint for the recovery may be imposed such as a time limit or maximum number of tries. If the internal state indicates the host is operating normally, and not attempting to recover, it sets its internal state to indicate that it is attempting to recover (block 825). At block 835, the host waits for the client host (e.g., Host A 100) to reconnect the receive socket, e.g.,

socket 172. If the reconnect attempt succeeds before a timeout (block 840), recovery is successful, and the host sets its internal state to indicate it has resumed normal operation (block 855). If the reconnect attempt fails, the host closes the server-initiated receive socket (block 845) and sets its internal state to indicate that it is disconnected from the second host (block 850).

[0025] The invention has been described herein with reference to particular exemplary embodiments. Certain alterations and modifications may be apparent to those skilled in the art, without departing from the scope of the invention. The exemplary embodiments are meant to be illustrative, not limiting of the scope of the invention, which is defined by the appended claims.